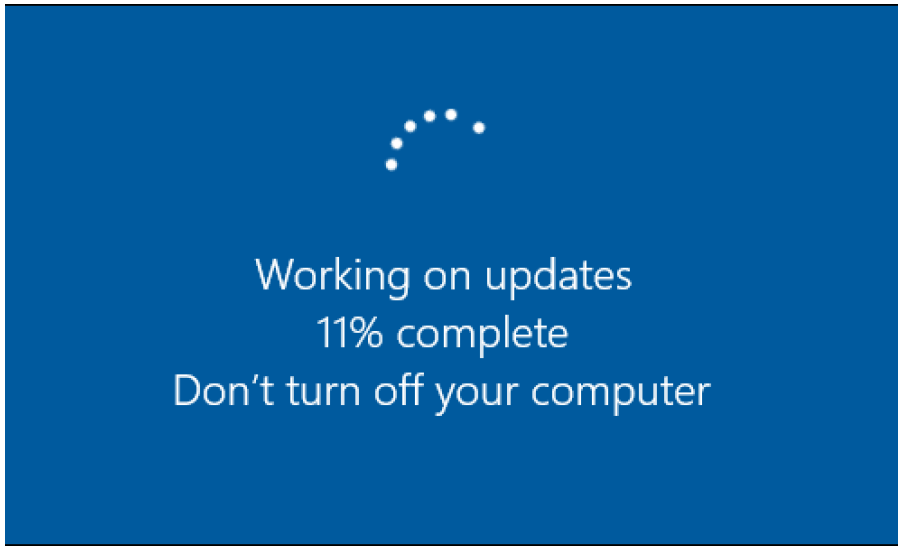
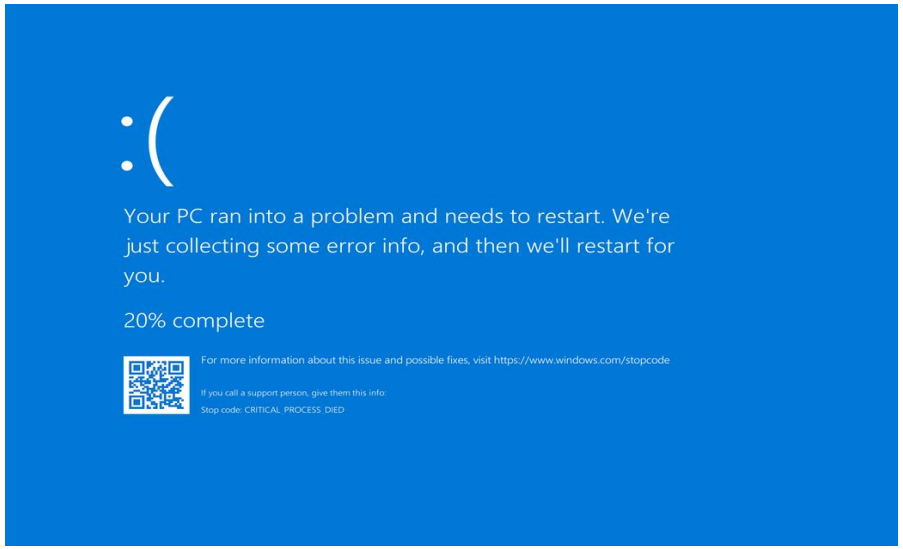
Reverse Debugging of Kernel Failures in Deployed Systems

Xinyang Ge, Ben Niu and Weidong Cui
Microsoft Research

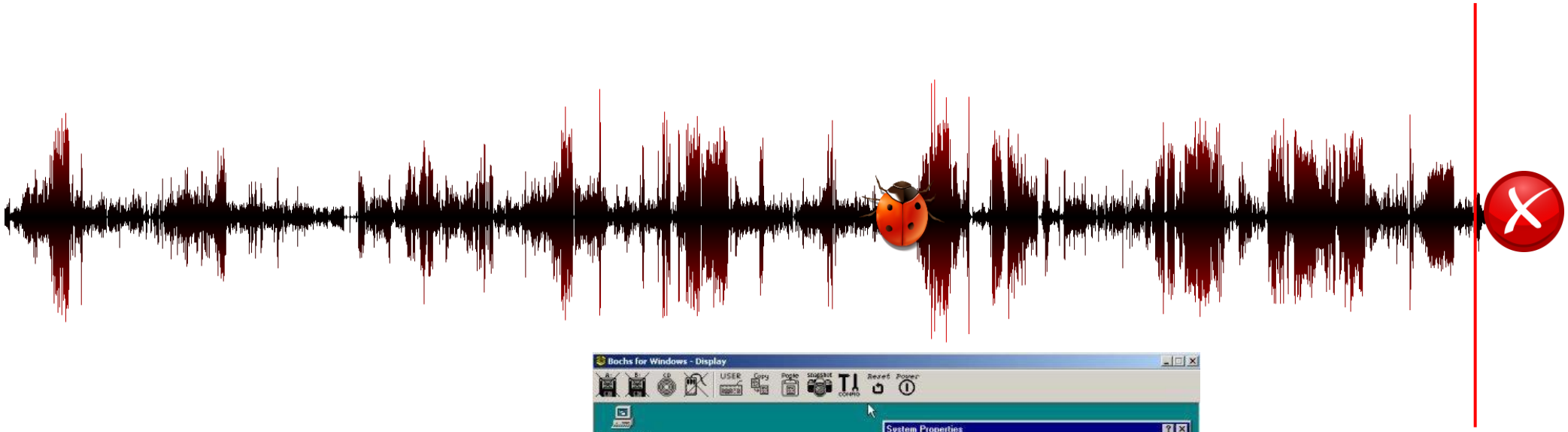
USENIX Annual Technical Conference, 2020


What happened
before the crash?
10110
01001
DMP

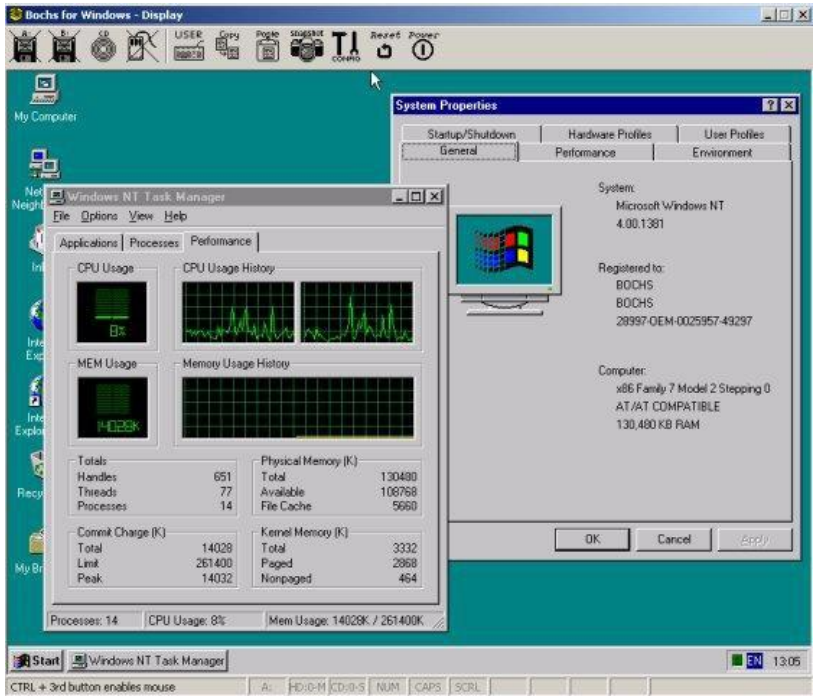

Working on updates
11% complete
Don't turn off your computer


:(
Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.
20% complete
For more information about this issue and possible fixes, visit https://www.windows.com/stopcode
If you call a support person, give them this info:
Stop code: CRITICAL_PROCESS_DIED


Windows
Most used
Get Started
Get Skype
Paint
Notepad
Sticky Notes
File Explorer
Settings
Power
All apps
New
Essentials
Apps
Office
Windows Insider
Ask me anything




A
HIGH QUALITY SOUND
X60


Bochs for Windows - Display
My Computer
System Properties
Startup/Shutdown
Hardware Profiles
User Profiles
General
Performance
Environment
System:
Microsoft Windows NT
4.00.1381
Registered to:
BOCHS
BOCHS
28997-OEM-0025957-49297
Computer:
x86 Family 7 Model 2 Stepping 0
AT/AT COMPATIBLE
130,480 KB RAM
OK
Cancel
Apply
Windows NT Task Manager
File Options View Help
Applications
Processes
Performance
CPU Usage
CPU Usage History
MEM Usage
Memory Usage History
Totals
Handles 651
Threads 77
Processes 14
Physical Memory (K)
Total 130480
Available 108768
File Cache 5660
Commit Charge (K)
Total 14028
Limit 261400
Peak 14032
Kernel Memory (K)
Total 3332
Paged 2868
Nonpaged 464
Processes: 14
CPU Usage: 8%
Mem Usage: 14028K / 261400K
Start
Windows NT Task Manager
EN
13:05
CTRL + 3rd button enables mouse

REPT: Reverse Execution with Processor Trace

REPT: Reverse Execution with Processor Trace

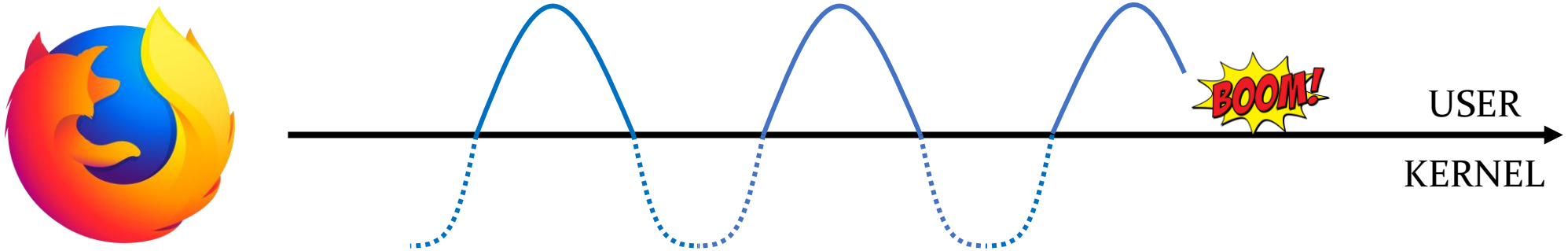
- A practical reverse debugging solution for user-mode failures [OSDI'18]

- Online hardware tracing (e.g., Intel Processor Trace)
  - Log the control flow with timestamps
  - Low runtime overhead (1-5%)
  - *No data!*

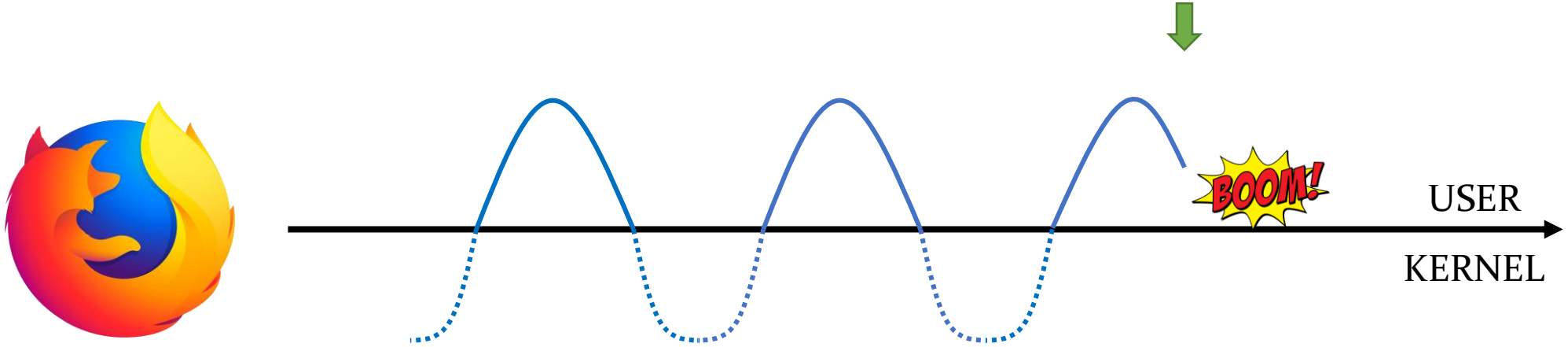
- Offline binary analysis
  - Recovers data flow from the control flow

How to make REPT support the kernel?

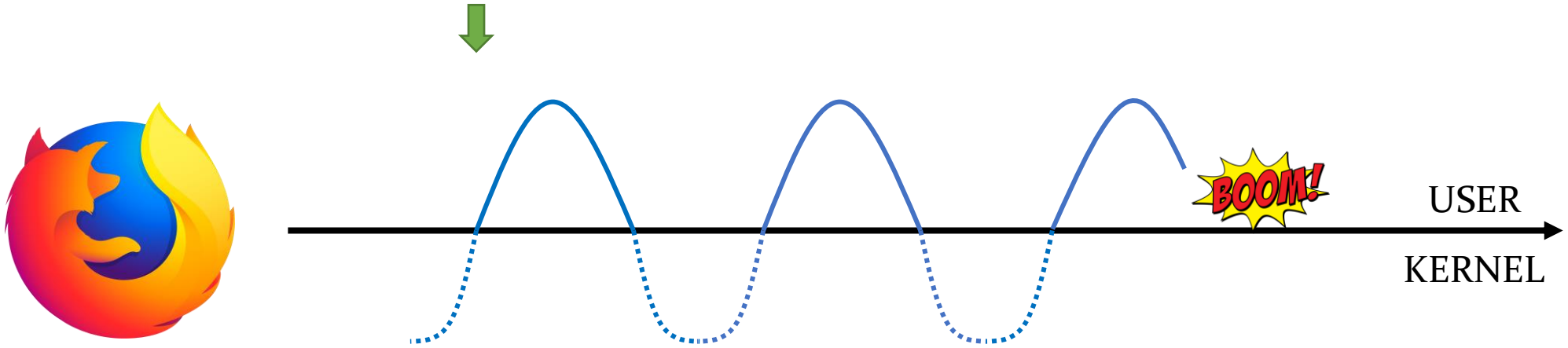
How REPT works?



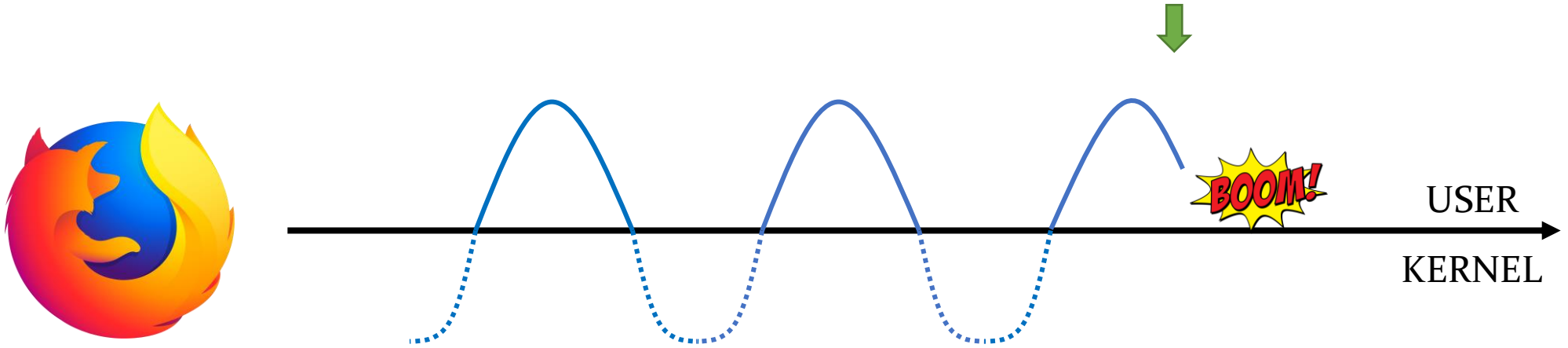
How REPT works?



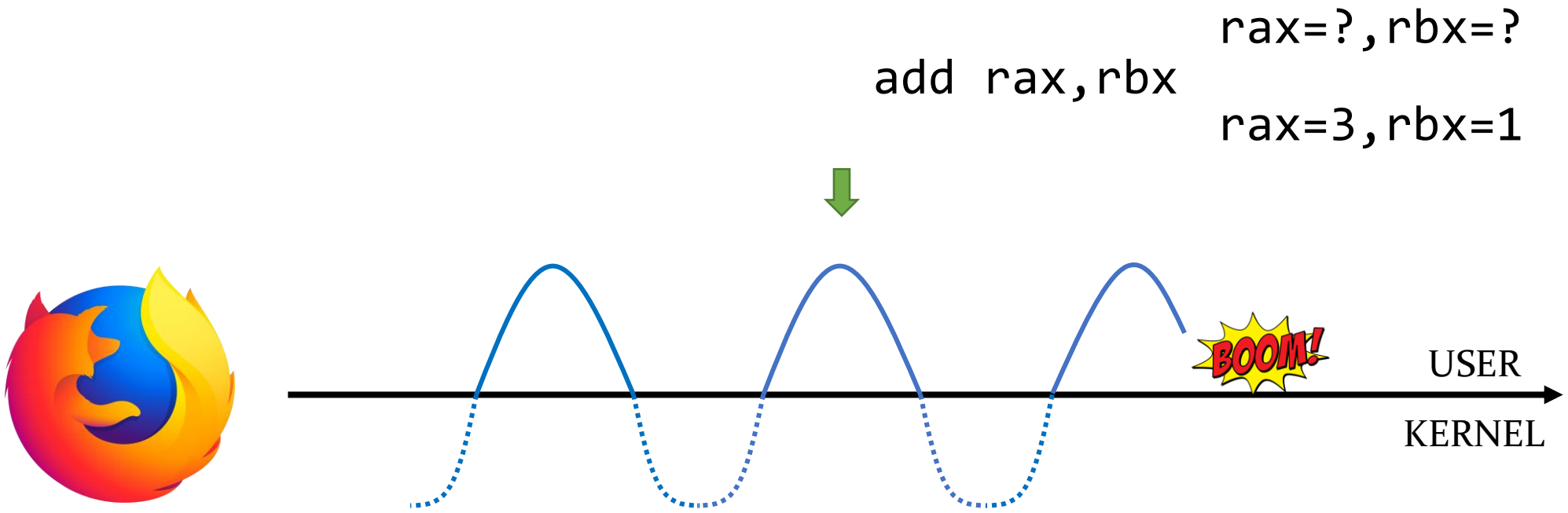
How REPT works?



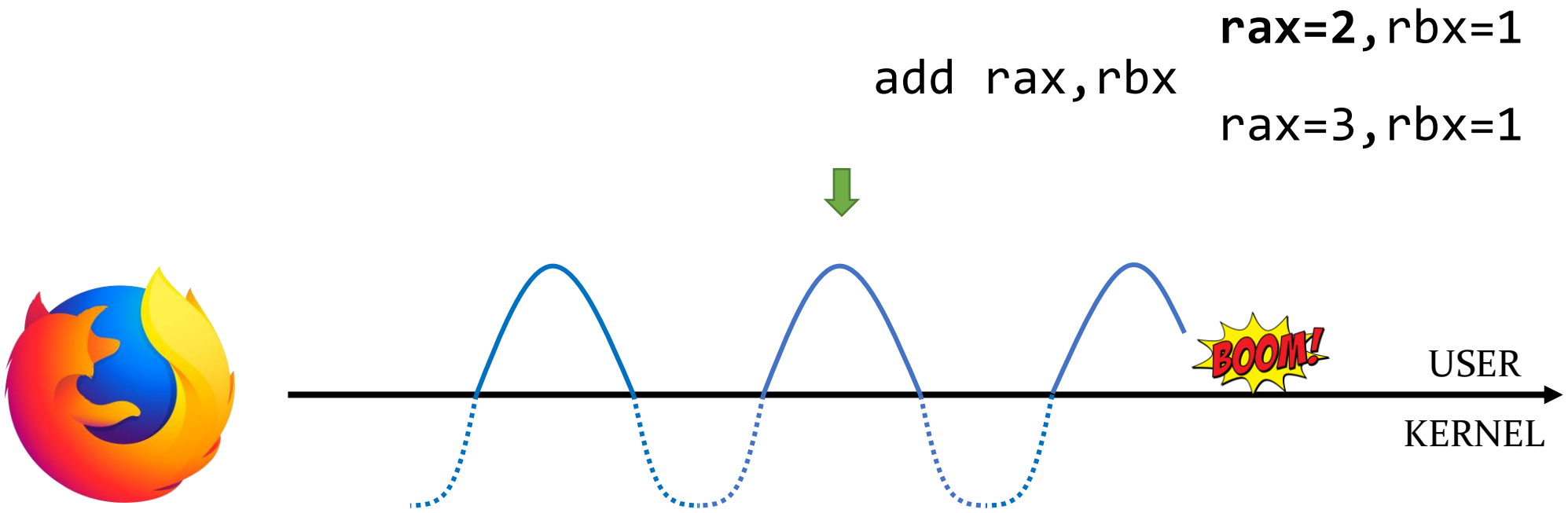
How REPT works?



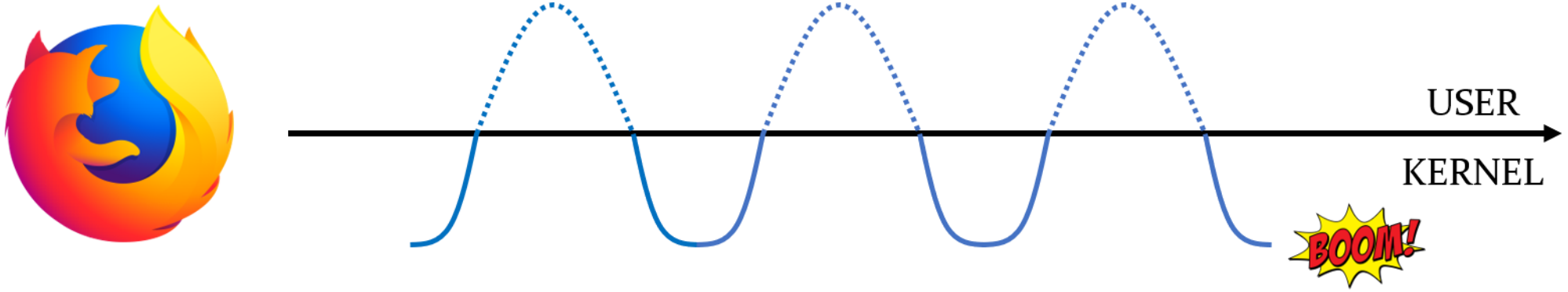
How REPT works?



How REPT works?



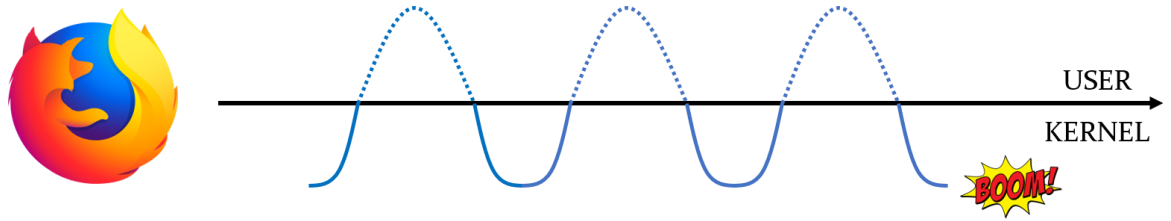
Can we simply inverse the tracing?



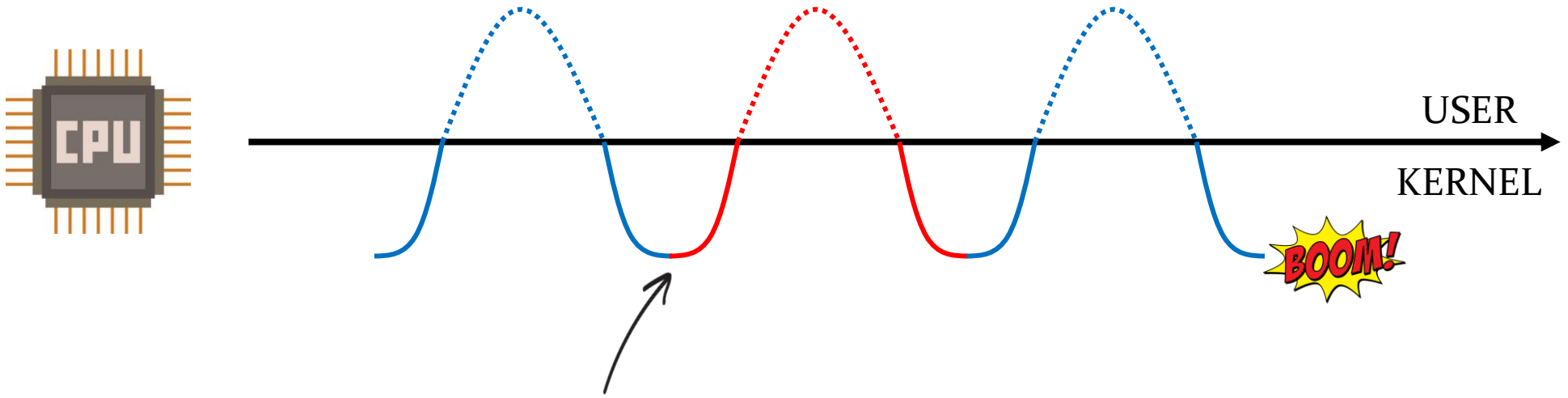
Can we simply inverse the tracing?

- There are too many processes/threads on a system
  - High memory overhead for tracing

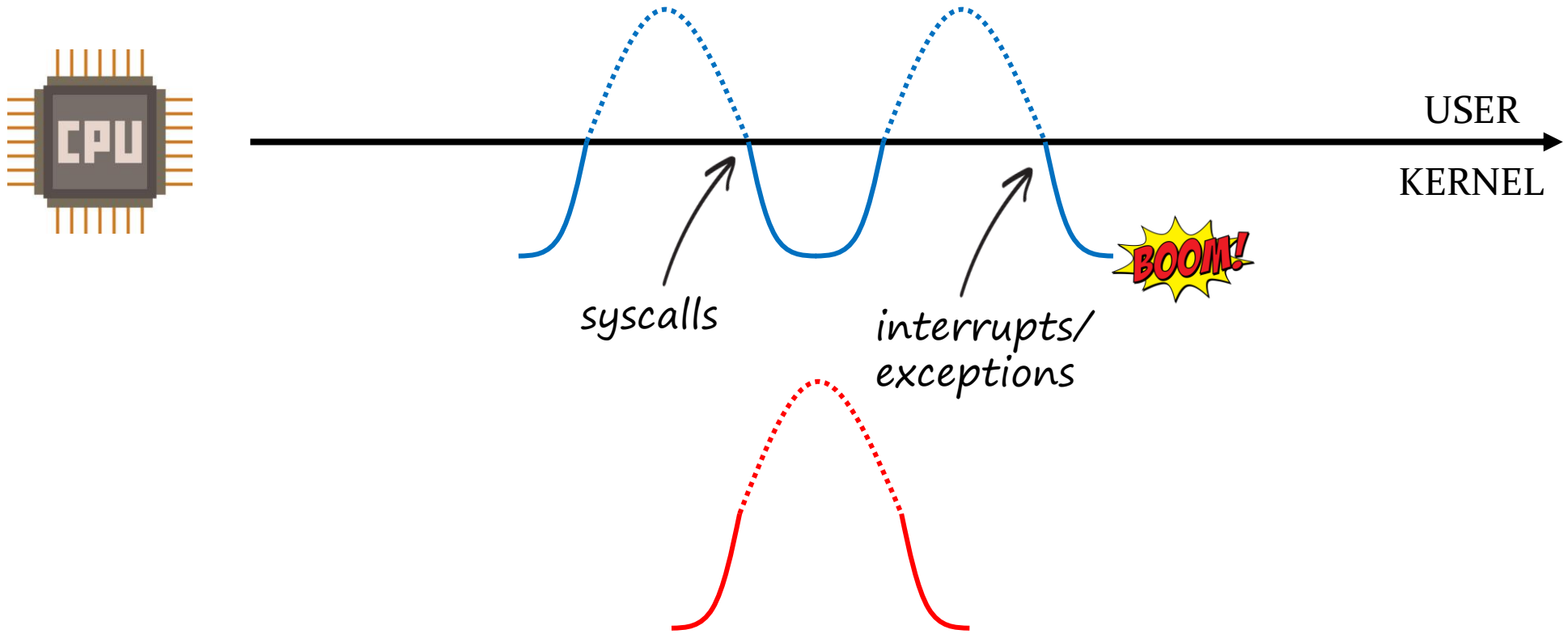
- Hardware events must be emulated in addition to CPU instructions
  - Interrupts
  - Exceptions
  - System calls

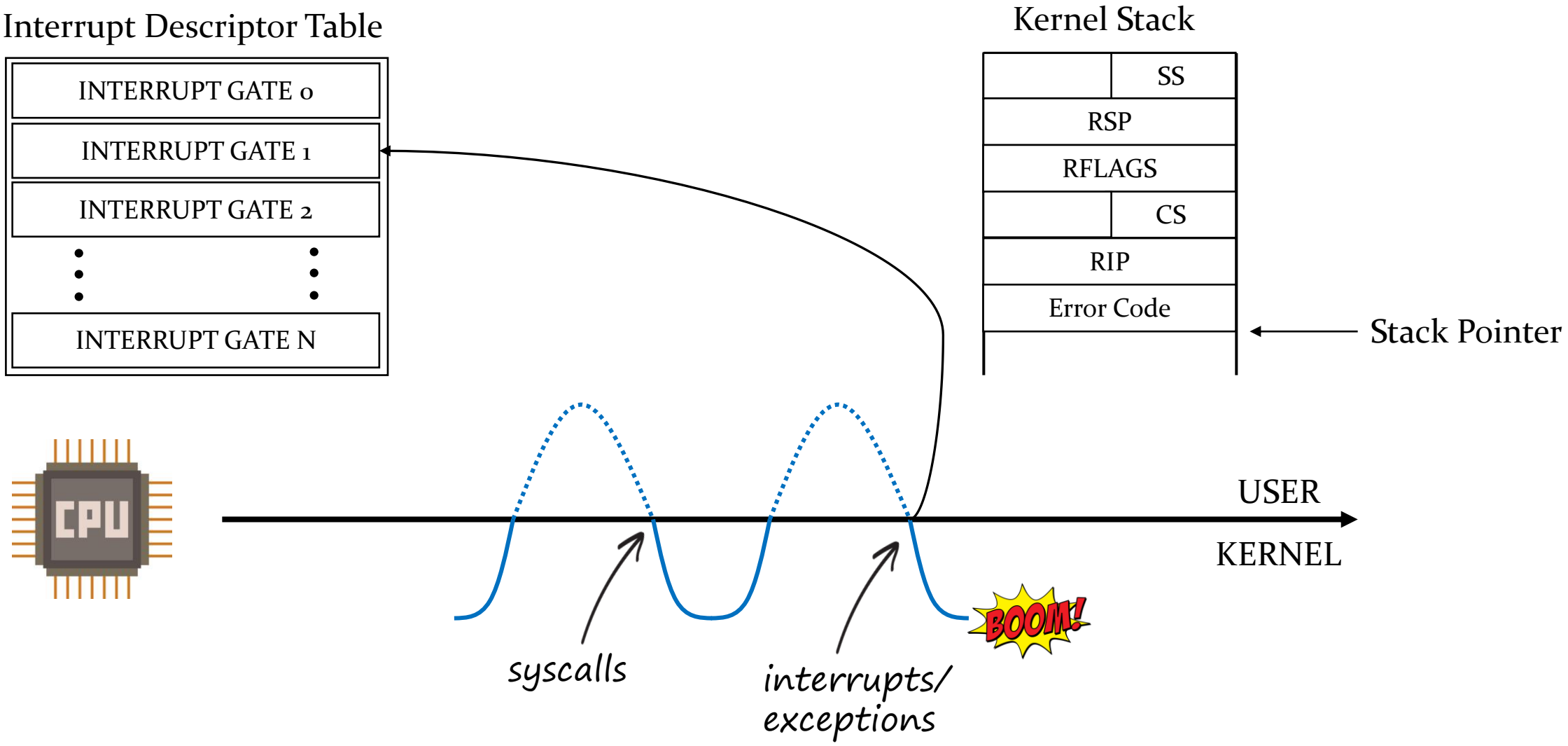


Here comes Kernel REPT…



... is irreversible, and we log it in software.


CPU
USER
KERNEL
BOOM!
syscalls
interrupts/
exceptions



Different events can have
different architectural effects

That’s it?

Automated Analyses

- A common bug pattern: missing undo operations
  - EnterCriticalRegion vs LeaveCriticalRegion
- Root-Cause Analysis
  - Scan the kernel execution trace to find missing undo operations
- Proactive Bug Detector
  - Sanitize the kernel execution based on specified invariants
  - 17 new bugs found and fixed!

Demo

Conclusion

- Debugging production kernel failures is hard

- REPT now supports the reverse debugging of the kernel
  - Per-core control flow tracing in hardware
  - Context switch logging in software
  - Recovers data flow via CPU instruction and hardware event emulation

- REPT enables automated analysis beyond reverse debugging
  - Root-cause analysis
  - Sanitizing analysis